CHAPTER 1

Introduction

"[. . .] Nothing at all takes place in the universe in which some rule of maximum or minimum does not appear."

- Leonhard Euler

The field of mathematical optimization has a long and storied history spanning multiple centuries. Dating back to the 1700s, famous mathematicians such as Newton, Fermat, and Lagrange each contributed significantly to the early development of mathematical analysis in finding optima of functions. In recent decades, pioneering works by physicists and mathematicians such as Richard Feynman, David Deutsch, Hidetoshi Nishimori, and Edward Farhi have created exciting new related fields of research: quantum computing and quantum optimization. Here we give a brief history and introduction to the core concepts in these fields, how they relate to combinatorial optimization, and the motivation for the research presented in this thesis.

Now known as "The Seven Bridges of Königsberg", this problem has become synonymous with the birth of the field of graph theory, and is stated as follows: is it possible to traverse the city of Königsberg so that each of the seven bridges is crossed once, and only once? Through trial and error, it is easy to demonstrate that such a walk is not possible for the city of Königsberg and its bridges, Fig. 1.1 (a). However, it was a peculiar curiosity from Leonhard Euler about this problem that spawned completely new branches of mathematics: graph theory and combinatorics. Euler's insight was that there were some innate topological features about the structure and connectivity of the land masses in the city which made such a walk impossible. The notion of a *graph* was invented: each land mass in the city can be represented by a *node*, and the bridges connecting them using *edges*, with the

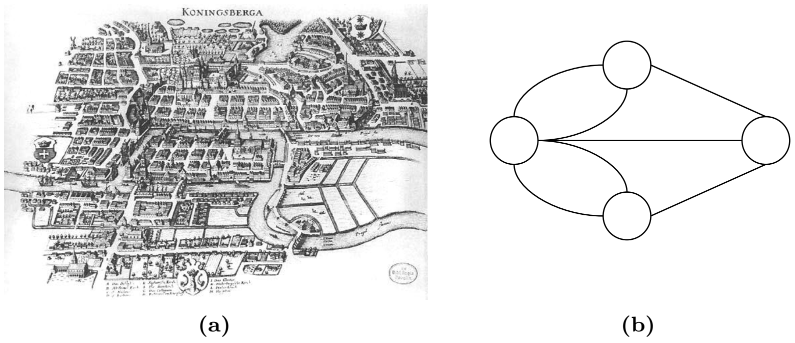

Figure 1.1: **(a)** Depiction of the city of Königsberg in Euler's time with its original seven bridges, circa 1700. Image is public domain, taken from Wikipedia Commons [3]. **(b)** Graph representation of the city of Königsberg, with land masses as nodes and bridges as edges.

number of edges connected to each node being its *degree*. Thus, it was possible to study the features of these graphical structure in an abstract way, rather than any one particular city (shown in Fig. 1.1 (b)). For Königsberg, the solution is that in order to have a walk that crosses each bridge only once, it may only have two odd-degree nodes; one for the start of the walk, and the other for its end. However, as shown in Fig. 1.1 (b), all four nodes in the graph have odd degree, and therefore no valid walk is possible [1]. Euler generalized this notion (later proven rigorously by Carl Hierholzer [2]), asserting that all graphs that have only even degree nodes have such a walk, and that in order for these walks to exist, there must be exactly zero or two nodes of odd degree in the graph. These are now named *Eulerian paths* in his honor.

While relevant historically for its role in graph theory, "The Seven Bridges of Königsberg" is not particularly interesting (or difficult) computationally. For every graph G with nodes V and edges E, Hierholzer's algorithm can find Eulerian paths in time $O(|E|)$. However, the problem serves as a conceptual intersection point between multiple disciplines that are central to this thesis. First is the connection between graph theory and combinatorics. While finding a Eulerian path is simple, and clearly relates to the theory of graphs and their connectivity, a simple related task of finding *all* Eulerian circuits in a graph is more difficult– it now deals with combinations and counting rather than pure graphical structure. In this thesis, similar such connections are presented (although undoubtedly less profound): the results of investigating the link between structures in graphs and the ability of quantum algorithms to solve problems based on them. Secondly, it is known that many optimization problems are fundamentally intractable at large sizes using classical computers, and so solving them may not be possible on practical timescales. Quantum algorithms offer a potential path towards solving these problems more efficiently. Analogously to the graph theory representation of bridges for Euler, we regard quantum algorithms as tools, and the research presented here reports on their use in the context of combinatorial optimization. The conceptual links that need to be made between optimization problems and quantum algorithms is why quantum optimization (and specifically quantum annealing) was investigated in this thesis. Lastly, it is important to appreciate the context in which Euler's original results were reported. While the original article was published in a scientific journal [1], the description of both the problem and solution relates to a problem that manifests in reality: the arrangement of bridges in a city. Therefore, the consequences of the theoretical properties proven in the paper could be observed directly by walking through the city of Königsberg and attempting to cross all bridges only once[1]. Essentially, the Königsberg bridges result could be regarded as a real-world application of graph theory. This intersection between theory and reality is also a cornerstone of the work in this thesis. The goal of the research that will be presented was to gain insight into how algorithms in quantum optimization may be used in practice, and therefore the motivation for many of the results are grounded in the application of these methods for real-world problems.

1.1 Foundations of quantum computing

The theory behind quantum computing (QC) dates to the early 1980s, developed independently by scientists Yuri Manin [4], Paul Benioff [5, 6] and Richard Feynman [7]. Together, these works spawned the fields of modern-day quantum computing and quantum information science. Yuri Manin was the first to envision such a paradigm as a more powerful computation model. In the works by Benioff, the focus was on the representation of a reversible universal Turing machine described by the evolution of a quantum system. The contribution from Feynman was

[1]Some of the original bridges have since been destroyed; today only two of the original seven remain [3]

concerned with the ability of classical computing machines to simulate quantum physics, a computationally intractable task. It was therefore proposed that one could design a programmable quantum system to perform computation. In [8], David Deutsch formalized the notion of a fully quantum model of computation, and specifies the theoretical concepts required to realize a universal quantum computer, $\mathcal{Q}$, the quantum Turing machine. These machines use the quantum equivalent of bits, referred to as quantum bits, or *qubits*. As with classical bits, these can be in either of the binary states 0 or 1. However, as quantum mechanical objects, qubits can also be in states which are combinations of both 0 and 1 simultaneously until measured (or observed). The states of observation are referred to as the *computational basis states*. Computational operations are defined via the use of unitary operators acting on these qubits (in discrete time steps) in order to construct quantum circuits, representing quantum algorithms (or quantum simulations).

In practice, the power of qubits (and therefore quantum computers) exploits three fundamental principles of quantum mechanics.

- **Coherence:** Each of the individual qubits in the quantum system cannot be described via classical physics, but can only be described by the time-dependent Schrödinger wave equation. Thus, the evolution of qubits in a quantum computer is described via quantum mechanical wave dynamics and unitary operations.
- **Entanglement:** The individual components of the quantum system are inseparable; meaning, they cannot be described independently. Only a single physical description of the ensemble of elements (e.g., the whole entangled system) is possible. Consequentially, any operation applied to a qubit in a quantum computer affects all other qubits that are entangled with it.
- **Superposition:** Each qubit in the quantum computer may be in multiple states at the same time. Observing the qubits' states results in the measurement of a single configuration based on the states' individual probabilities. Furthermore, the observation (or collapsing) of the qubits results in the destruction of the other states not measured.

Using these concepts, quantum algorithms have been developed and discovered over the years, some of which significantly outperforming their classical counterparts. In [9], the first such algorithm was presented, now known as the Deutsch-Jozsa algorithm. Other famous examples include Shor's algorithm for factoring prime numbers [10], and Grover's algorithm for unstructured search [11].

Despite the theoretical algorithmic promises presented above, the field of quantum hardware for computation, however, has been slow to progress at the same pace. The first examples of experimental qubits involved NMR technology, where small qubit systems were built to execute simple quantum algorithms [12, 13, 14]. These prototypes were followed by similar experiments with trapped ion qubits [15], and soon after in superconducting flux qubits [16]. Since then, many other possible qubit technologies have been demonstrated for quantum computing, such as diamond cavities [17], quantum dots in silicon [18], cold atoms [19, 20, 21], and photonic quantum computing [22]. However, of these, currently superconducting qubits have proven to be the most scalable architecture choice, with companies such as Google [23], IBM [24], and D-Wave Systems [25, 26] building publicly-accessible platforms to access their quantum processing units (QPUs). These processors have been described as representative of the *noisy intermediate-scale quantum* (NISQ) era [27], describing larger, albeit still noisy (non-error-corrected) qubit systems. Each of these QPUs require different considerations specific to the design and implementation choices made during manufacturing, making them potentially more or less useful under certain conditions. However, they all use the same mathematical formulation and notation in order to execute quantum algorithms. The mathematical representation of the optimization problems presented in this thesis and the quantum algorithms used to solve them are presented next and will remain consistent for the remainder of this thesis.

1.2 Definitions and notation

1.2.1 Problem complexity

Combinatorial optimization is defined as selecting an optimal subset from a finite set of elements, usually subject to some objective. The work presented in this thesis is mainly focused on the tasks of solving such optimization problems using quantum optimization algorithms, in particular the quantum annealing algorithm. Conventionally, optimization problems are defined by their *problem class* (i.e., the type of problem) within a *complexity class* (i.e., complexity of the problems). These complexity classes are defined by evaluating the efficiency of solving specific problems using Turing machines (typically denoted using the symbol $\mathcal{T}$). For example, the aforementioned problem of finding a Eulerian path is solvable by a Turing machine in time polynomial in the graph size. However, a simply-stated related problem, the task of visiting every *vertex* exactly once in a given undirected graph, rather than its edges, currently does not have such a polynomial-time algorithm. Clearly, it is evident that by examining the efficiency of Turing machines in solving different combinatorial optimization problems (and related graph problems), different classes of problem complexity arise, and the field of studying the classification of problem using them is known as "complexity theory". A diagram with common classes based on Turing machines and their conjectured relations are shown in Fig. 1.2. The most pertinent complexity classes are defined briefly here:

- **P.** This problem class is defined as all decision problems which are solvable by deterministic Turing machines with resources (runtime) that scale polynomially with the size of the instance being solved. Typically, problems in this class are referred to as having "efficient" algorithms to solve them, and so efficiency is often equated to polynomial-time algorithms (also known as tractable problems).

- **NP.** The class of decision problems in which the verification of the correctness of a solution to a decision problem can be performed in polynomial time by a Turing machine. Equivalently, this problem class can be defined as the set of all decision problems which are solvable in polynomial time by *non-deterministic* Turing machines. Problems which can be used to represent all other problems in this class (with at most polynomial overhead) form a related class, known as **NP-hard**. Furthermore, problems which are NP-hard and themselves are in NP are referred to as **NP-complete**. It is important to note that this is not restricted to decision problems, and so many NP-complete decision problems have optimization versions which are NP-hard.

- **APX.** This class is defined as the set of NP optimization problems which have a polynomial-time approximation algorithm with bounded constant error. This is not to be confused with the set of optimization problems which have polynomial-time approximation schemes (PTAS), which is defined as optimization problems with polynomial-time approximation algorithms for *every* constant error bound (and thus are all contained within APX).

Furthermore, as in NP, there are APX-hard and APX-complete classes, defined as all problems that have approximation-preserving reductions, and problems which are both APX-hard and in APX, respectively.

Although Turing machines have been traditionally used because they are equivalent to every other model of computation, this does not necessary imply anything in regards to the efficiency of all models of computation. More recently, with the discovery of quantum algorithms and the invention of quantum computation, other problem classes have been developed, defined by the efficiency of quantum algorithms in solving them. These are defined by considering the existence of quantum Turing machines, quantum mechanical extensions of the original Turing machines. A diagram depicting the relationships between the quantum problem classes is in Fig. 1.2. The relevant quantum complexity classes are defined here:

- **BQP.** This class is defined as the set of decision problems which can be solved in polynomial time by a quantum Turing machine with an error probability bounded by 1/3. Since every classical circuit can be simulated with quantum circuits (and can be constructed efficiently), P is contained within this class.
- **QMA.** This is the class of problems which have a polynomial-time quantum circuit that can verify a given polynomial-size quantum state (or proof) with bounded error. As with NP, the set of problems to which all QMA problems are reducible to are known as **QMA-hard**, and those that are both QMA-hard and in QMA themselves are known as **QMA-complete**.

The relationships between many of the complexity classes defined above, both classical and quantum, are still unknown. For example, the question of whether NP is contained in P (referred to as $P \stackrel{?}{=} NP$) is an open fundamental question in computer science (first introduced in [28]), the consequences of which would collapse several of the sets shown in Fig. 1.2. Furthermore, the relations between the quantum complexity classes to the classical ones are also not fully known. For instance, although P is contained in BQP, there are problems solvable by BQP which are conjectured to not be in P. The most well-known example of this is the problem of finding prime factors, which is in NP and has a polynomial-time quantum algorithm (known as Shor's factoring algorithm [10]). Thus, there are still exciting and potentially ground-breaking theoretical results to be discovered.

The ultimate goal of the work presented in this thesis is not to settle these fundamental (and important) theoretical questions. Rather, the context of the research

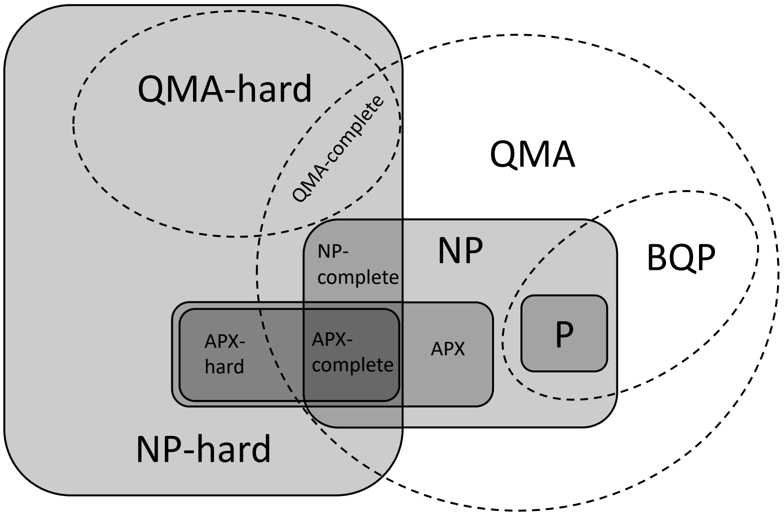

Figure 1.2: A diagram of complexity classes and their relationships, as explained in the text. Classical complexity classes are in rounded squares, and quantum complexity classes are in dashed ellipses.

was to assess the practical relevance of a specific class of quantum optimization algorithms (quantum annealing) implemented on state-of-the-art quantum hardware. The complexity classes defined above were used as a guide in order to better understand what conditions need to be fulfilled, and which problems need to be investigated, to meet such usefulness criteria.

1.2.2 Qubits and operators

The basic elements with which quantum algorithms are built are qubits and their respective unitary operators. The two qubit states which are used to define the vector space of operations are:

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}, \quad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}. \tag{1.1}$$

These two states together form an orthonormal basis, called the *computational basis states.* In Dirac notation[1], the single qubit state (typically denoted as the wavefunction ψ) is represented using a linear combinations of the basis states:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle . \tag{1.2}$$

Here, α and β, which are complex numbers, are the respective amplitudes of each basis state. Consistent with the theory of quantum mechanics, one can only observe each state with a finite probability. As such, the values of the amplitudes are constrained by:

$$|\alpha|^2 + |\beta|^2 = 1. \tag{1.3}$$

Analogous to classical computation, these qubits are manipulated through the use of operators (or gates). In quantum mechanics, unitary operators are used, represented as matrices operating on the vector space of qubits. The central operators for quantum annealing algorithms are the Pauli spin matrices:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{1.4}$$

In quantum annealing, the Pauli operators are applied to sets of qubits in the quantum system in order to encode the optimization problem, and their magnitudes are varied throughout the computation to perform minimization.

1.2.3 Ising models and QUBO problems

Depending on the specific problems being solved, it is convenient to specify different bases for the target optimization problem. Some problems may use variables $s \in \{-1, 1\}$ called *spins*, while others may use binary variables $x \in \{0, 1\}$. These two are equivalent via a simple change of basis variables, $s = 2x - 1$. Objective functions with spin variables are represented as Ising spin-glass systems (also known as Ising models):

$$\mathcal{H}(s) = \sum_i h_i s_i + \sum_{i<j} J_{ij} s_i s_j, \tag{1.5}$$

[1] Dirac notation (also known as bra–ket notation) is a method to describe the linear algebra governing quantum mechanics. The derivation and background of this notation is beyond the scope of this thesis, but an introduction to the topic can be found in [29]

where h_i is the linear weight (also known as bias) of each spin variable s_i, and J_{ij} is the quadratic interaction term (also known as coupling strength) for each spin pair. Equivalently, one could specify the objective function using binary variables as a quadratic unconstrained binary optimization (QUBO) problem:

$$\text{Obj}(x, \mathbf{Q}) = x^T \cdot \mathbf{Q} \cdot x. \tag{1.6}$$

Here, x is a vector of N binary variables, and $\mathbf{Q}$ is an $N \times N$ symmetric interaction matrix. Using the property $x_i \cdot x_i = x_i^2 = 1$ for binary variables, the QUBO objective can be re-written in the same form as the Ising model, with the diagonal elements of $\mathbf{Q}$ being the linear terms, and the off-diagonal elements being the quadratic terms[1]. The task performed by the quantum optimization algorithm is to minimize the objective functions above.

1.3 Families of combinatorial optimization algorithms

Given the complexity of many hard combinatorial optimization problems, various algorithms have been developed in an attempt to solve them, both to optimality and approximately. In this thesis we are concerned with specific kinds of algorithms in order to both test quantum annealing directly and to contextualize the results via comparisons to other algorithms. We briefly define and explain the different families of algorithms below.

Exact algorithms. Algorithms which have deterministic endpoints that are guaranteed to provide optimal solutions to problems are known as exact algorithms. Because of the open question of P vs. NP, currently all exact algorithms for NP-complete and NP-hard optimization have exponential runtime in the worst case. Nonetheless, a variety of exact algorithms are used for real-world combinatorial optimization problems, such as CPLEX [30] and Gurobi [31], among others.

Classical (meta)heuristics. Algorithms implemented on classical computers which exploit probabilistic (or non-deterministic) routines within them to solve specific problems are referred to as *heuristics*. More general algorithms which use generic (and often parameterized) routines to solve arbitrary optimization problems

[1]Describing the matrix $\mathbf{Q}$ as an upper- or lower-triangular matrix is also common, since all terms $x_i \cdot x_j = 0$ if either x_i or x_j are 0.

are known as *metaheuristics*. Since (meta)heuristics can solve some instances of problems in NP, they are often used as benchmarks in comparative analyses. Examples of well-known metaheuristics are genetic/evolutionary algorithms [32], simulated thermal annealing [33], particle swarm optimization [34], and many more.

Quantum (meta)heuristics. Equivalently, one can define heuristics and metaheuristic algorithms for quantum computers, which in turn implement probabilistic routines in quantum circuits or Hamiltonians. The quantum annealing algorithm is one such metaheuristic for quantum optimization [35]. The quantum approximate optimization algorithm (QAOA) is a similar quantum optimization metaheuristic for gate-model quantum computers [36].

Hybrid quantum-classical algorithms. Specifically in the near-term era for quantum hardware, the limiting factor in implementing quantum algorithms is often the number of available qubits. As a consequence, the concept of hybrid algorithms has been developed, in which quantum algorithms are used as an inner loop to perform a specific task set by a classical algorithm acting as an outer loop. There are many possible ways to construct such quantum (meta)heuristics, both for quantum annealers and gate-model quantum computers [37, 38, 36, 39]. In the context of this thesis, we refer to any classical algorithm which offloads a sub-task to an adaptive quantum inner-loop as a *hybrid algorithm*, to distinguish from more traditional parameter-tuning techniques.

1.4 Outline

We now outline the remainder of the work in this thesis. Each of the chapters below represents a summary of one or more individual projects addressing the respective research question motivating the research. To address the overall goal of applying quantum annealing to industrial combinatorial optimization problems, we break the topic down into a series of main research questions, each motivating and expanding upon a different aspect in the topic. The chapters are built upon peer-reviewed publications, listed at the end of the description of each chapter.

Q1. How are combinatorial optimization problems expressed for quantum annealers?

Quantum annealers are special-purpose quantum hardware which implement the quantum annealing algorithm to perform optimization, and share some similarities to classical metaheuristic optimization algorithms. However, the process required to perform optimization using quantum annealing has multiple steps which have no analogy in classical optimization. The impact of these steps on the hardware's ability to solve problems is not obvious. To answer **Q1**, Chapter 2 presents the theoretical motivation for the quantum annealing algorithm, and the particularities of its manifestation in hardware. Then, the steps required in order to formulate and submit optimization problems to quantum annealers in practice are reviewed.

[1] S. Yarkoni, E. Raponi, T. Bäck, and S. Schmitt (2022). Quantum Annealing for Industry Applications: Introduction and Review. *Rep. Prog. Phys.* **85** 104001.

Q2. What is the impact of hyperparameters, tuning, and physical limitations of quantum hardware on algorithmic performance?

Using a representative canonical NP-hard problem (the maximum independent set problem), Chapter 3 explores the setting and tuning of QPU parameters. The trade-off between algorithmic runtime and performance with respect to this parameter tuning process is investigated. To answer **Q2**, best practices and algorithms developed to tune these parameters are implemented and tested on a D-Wave 2000Q QPU. Where relevant, the physics governing the QA algorithm are used to motivate the appropriate schemes for choosing and tuning parameters. The ability to solve optimization problems in practice using these methods is demonstrated. The results from the analysis allow general guidelines and conclusions to be drawn in regards to combinatorial optimization using QA.

[1] S. Yarkoni, T. Bäck, and A. Plaat (2018). First results solving arbitrarily structured Maximum Independent Set problems using quantum annealing. In *Proceedings of the 2018 IEEE Congress on Evolutionary Computation*, CEC '18, Rio de Janeiro, Brazil, pp. 1–6.

[2] S. Yarkoni, H. Wang, A. Plaat, T. Bäck (2019). Boosting Quantum Annealing Performance Using Evolution Strategies for Annealing Offsets Tuning. In: Feld S., Linnhoff-Popien C. (eds) Quantum Technology and Optimization Problems, QTOP '19. Lecture Notes in Computer Science, vol 11413. Springer, Cham, pp. 157–168.

Q3. How are real-world problems different from academic problems, and how does this affect the performance of quantum annealing?

Real-world optimization problems often require mixtures of either variable types, constraints, or other such terms which make them more difficult to represent relative to the canonical problems presented thus far. Chapter 4 introduces methods to formulate such combinatorial optimization problems for quantum annealing, with special consideration towards mathematical modeling of problems. This includes the addition of constraint-based optimization, and the impact such problems have on performance is investigated. The potential and limitations of quantum annealing for solving real-world optimization problems are explored using a variety of use-cases and datasets.

[1] Sheir Yarkoni, Andrii Kleshchonok, Yury Dzerin, Florian Neukart, Marc Hilbert (2021). Semi-supervised time series classification method for quantum computing. Quantum Machine Intelligence **3**, 12.

[2] Sheir Yarkoni, Andreas Huck, Hanno Schulldorf, Benjamin Speitkamp, Marc Shakory Tabrizi, Martin Leib, Thomas Bäck, and Florian Neukart (2021). Solving the Shipment Rerouting Problem with Quantum Optimization Techniques. In: International Conference on Computational Logistics (ICCL 2021): Computational Logistics, pp. 502-517.

Q4. Can we use hybrid quantum-classical to overcome some of the deficiencies of current quantum hardware?

Limited sizes of currently-available quantum hardware is a bottleneck in many application areas. Therefore, Chapter 5 utilizes the previously presented work, and introduces the concept of hybrid quantum-classical algorithms. Various techniques and methods for hybrid algorithms are both reviewed and implemented, developing the necessary functionality to build end-to-end applications with quantum annealing. These algorithms are testing using real-world data and live applications are built using quantum annealers and hybrid algorithms. The capabilities of quantum annealers and hybrid algorithms to solve industrial optimization problems are explored and presented.

[1] Sheir Yarkoni, Florian Neukart, Eliane Moreno Gomez Tagle, Nicole Magiera, Bharat Mehta, Kunal Hire, Swapnil Narkhede, Martin Hofmann. (2020). Quantum Shuttle: Traffic Navigation with Quantum Computing.

In: Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software (22-30).

[2] Sheir Yarkoni, Alex Alekseyenko, Michael Streif, David Von Dollen, Florian Neukart, and Thomas Bäck (2021). Multi-car paint shop optimization with quantum annealing. In: 2021 IEEE International Conference on Quantum Computing and Engineering (QCE): 35-41.

Finally, Chapter 6 discusses the future of quantum annealing for optimization, draws conclusions, and examines the lessons learned from the work presented.